Lab Exercise: Packages

INTERTECH

Instructors Who Consult. | Consultants Who Teach.

About Intertech

Thank you for choosing Intertech for your training from Udemy. The next page of this document will get you started with your lab if you'd like to skip ahead. Below is a brief overview of Intertech as well as a promo code for you for future live Intertech trainings.

Intertech (www.intertech.com) is the largest combined software developer **training** and **consulting** firm in Minnesota. Our unique blend of training, consulting, and mentoring has empowered technology teams in small businesses, Fortune 500 corporations and government agencies since 1991.

Our training organization offers live in-classroom and online deliveries, private on-site deliveries, and on-demand options such as the course in which you've enrolled from Udemy. We cover a broad spectrum of .NET, Java, Agile/Scrum, Web Development, and Mobile technologies. See more information on our training and search for courses by **clicking here**.

As a Udemy customer, you can save on any live in-classroom, live online, or onsite training with Intertech as well. Just use promo code "Udemy_Labs" when enrolling **and** save 35% on the course price.

We appreciate you choosing Intertech on Udemy for this course!

Lab Exercise

Packages

An application can be comprised of thousands of classes with dozens of programmers working on the code. Furthermore, third-party classes are often borrowed or purchased to reduce application development time. So how do you keep all that code organized? How do you prevent a third party's Account class from being confused with your Account class? The answer to these questions is packaging. In this lab, you organize the Acme Order System application through the use of packages.

Specifically, in this lab you will:

- Create three packages
- Move existing code to these packages
- See the use of package and import statements in the code
- Begin to explore the effect of packaging on access to data and methods in classes
- Optionally build a Java archive (JAR) of your existing code.

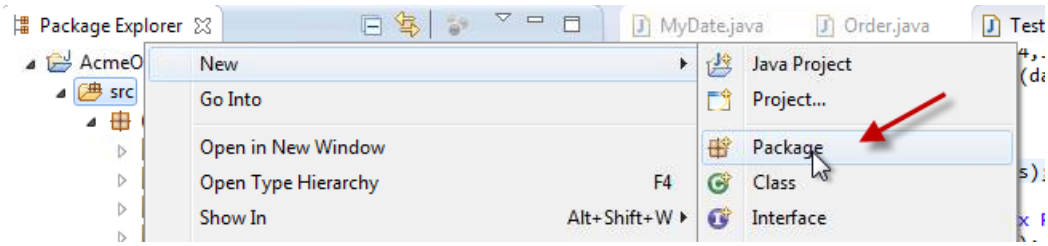
Scenario

The manager of the Acme order system had been pleased with the results of your work so far, but then he took a look at the source code organization and wasn't so happy! "Get that stuff organized!" he demands.

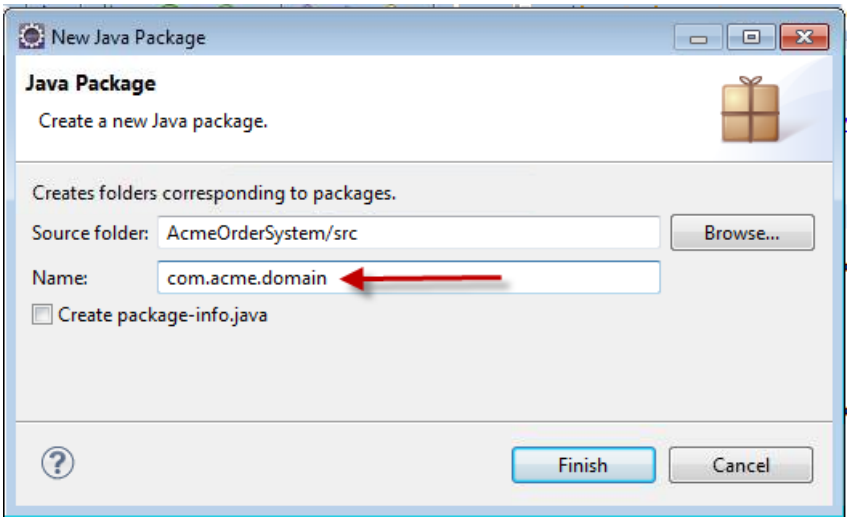
You decide to organize the code into several packages. Business classes like Order will go into a com.acme.domain package. Utility code, like the MyDate class, will go into com.acme.utils package. Finally, testing code will go into a com.acme.testing package.

Step 1: Create the packages in Eclipse

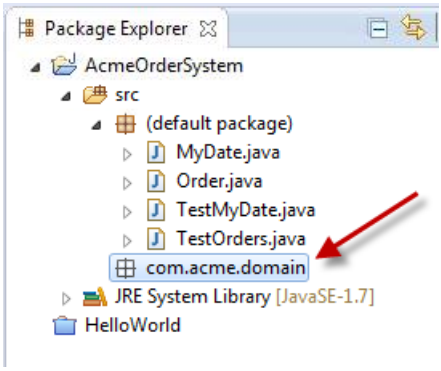
1.1 In the Package Explorer view, right-click on the src folder in the AcmeOrderSystem project and select ***New > Package***.



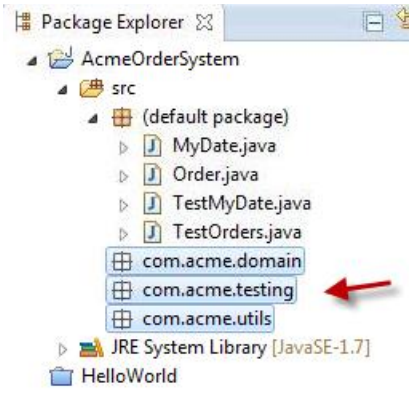
1.2 In the resulting New Java Package window, enter com.acme.domain as the package name for the new package and click the ***Finish*** button.



This should result in a new package displayed in the Package Explorer view.



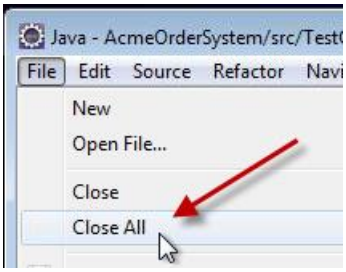
1.3 Create two other new packages (com.acme.utils and com.acme.testing) using the same process.



Step 2: Add classes to the packages

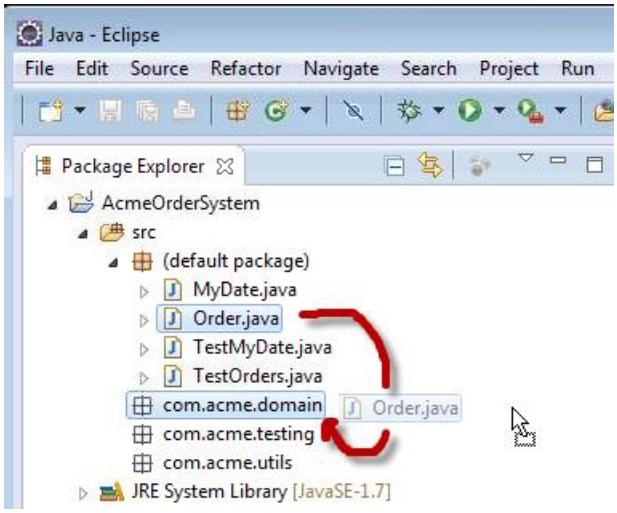
Notice that in the Package Explorer view, your existing classes are all in a single "(default package)". In this step, you move the classes to their appropriate package.

2.1 Close all open editors. From the Eclipse menu bar, select ***File > Close All***.

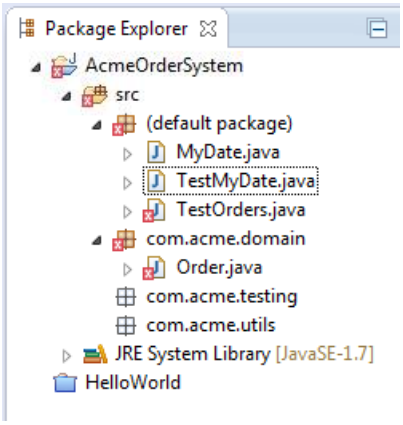


2.2 Move Order.java to the com.acme.domain package.

2.2.1 In the Package Explorer view, drag and drop Order.java into the com.acme.domain package.



2.2.2 This will cause the file to be listed as part of the com.acme.domain package.



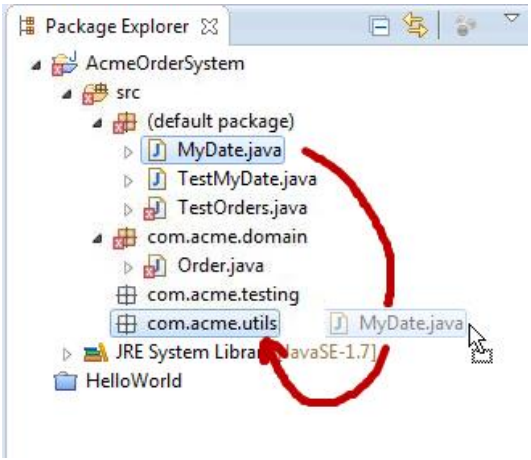
Note*:* Moving Order to the com.acme.domain package also creates compiler errors in other areas of your code! Do you know why? Don't worry, as you fix these problems later.

2.2.3 Open the Order.java file in an editor view by double-clicking on the file name in the Package Explorer view. What changes did this move create in your code? Notice the package and import lines of code at the top of the file.

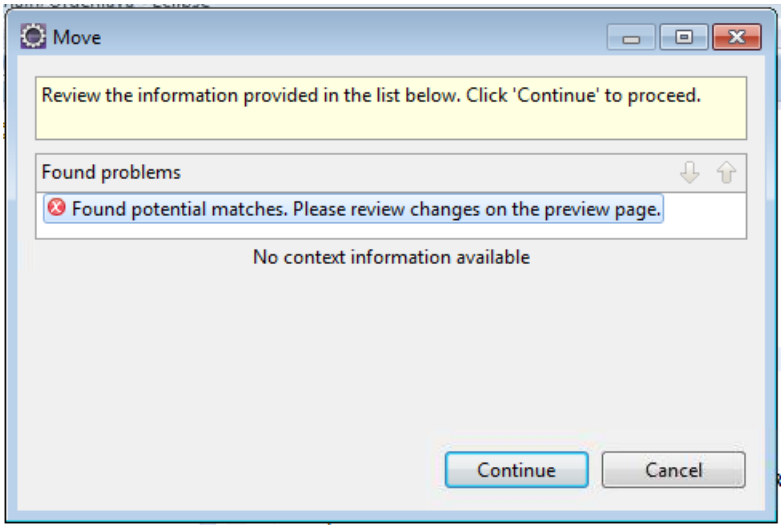
```
package com.acme.domain;
import MyDate;
```

2.3 Move MyDate.java to the com.acme.utils package.

2.3.1 Just as you moved Order.java, drag and drop MyDate.java into the com.acme.utils package.

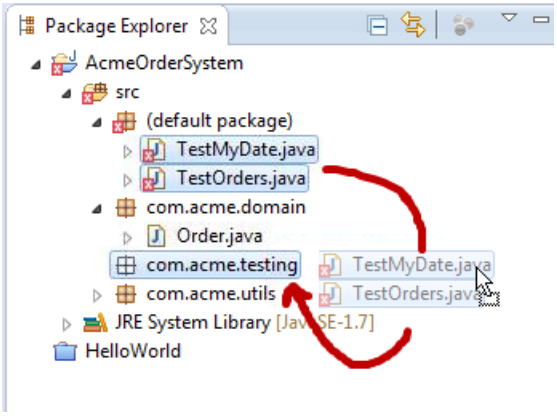


2.3.2 This time you should also be confronted with a warning window as shown below.



Eclipse is detecting references to MyDate in other code and is negotiating these issues. Click the *Continue* button to allow the move of the class into the package.

2.4 Move TestOrders.java and TestMyDate.java to the com.acme.testing package. As you moved the other classes, drag and drop the two remaining classes in the default package to the com.acme.testing package.



Note*:* With the last class moved, you may notice that the default package is removed from the Package Explorer view!

2.5 In an editor view, open Order.java, this is now found in the com.acme.domain package.

2.5.1 Notice that the Order.java file carries the line at the top that designates what package the Order type is a part.

```
package com.acme.domain;
```

2.5.2 Notice also, that it appropriately imports com.acme.utils.MyDate as this type is used as the type for one of Order's attributes (the orderDate).

```
import com.acme.utils.MyDate;
```

2.6 Open TestOrders.java (found in the com.acme.testing package) in an editor view.

2.6.1 Notice that TestOrders appropriately has the package indicator at the top of the file.

```
package com.acme.testing;
```

2.6.2 Notice that since TestOrders has to work with both MyDate objects and Order objects, it has two *import* statements.

```
import com.acme.domain.Order;
import com.acme.utils.MyDate;
```

Step 3: Resolve access issues

Some of the errors created in all the moving of classes to the different packages went away, but compiler errors in the test classes (TestOrders and TestMyDate) still remain. You must now fix these issues.

3.1 Open TestMyDate.java (found in the com.acme.testing package) in an editor view.

3.1.1 Notice that there are three compile errors (maybe more depending on what bonus labs you completed). What is the cause of these errors?

3.1.2 The errors have to do with the fact that the MyDate object fields (month, day, and year) are not public and therefore not accessible to TestMyDate once it moved to a different package.

3.1.3 You will cover access to data in objects a little later in this class. How do you fix these errors now? Go to the next step to see.

3.2 Open MyDate.java (now in the com.acme.utils package) in an editor view.

3.2.1 Change fields on MyDate to be public. That is, put the keyword "public" in front of the month, day, and year variables.

```
public int day;
public int month;
public int year;
```

3.2.2 This change should make the compiler errors in TestMyDate.java go away.

3.3 Open TestOrders.java in an editor view. Notice that the same type of errors exist in this test class for the static variable taxRate.

3.4 Open Order.java in an editor view. Make taxRate public too. This should make the other compiler errors go away.

```
public static double taxRate = 0.05;
```

Step 4: Run the tests

Run both TestMyDate and TestOrders to ensure all of your code is still working correctly after the package reorganization.

Lab Solution

Order.java

```
package com.acme.domain;
import com.acme.utils.MyDate;

public class Order {
  MyDate orderDate;
  double orderAmount = 0.00;
  String customer;
  String product;
  int quantity;

  public static double taxRate = 0.05;

  public static void setTaxRate(double newRate) {
    taxRate = newRate;
  }

  public static void computeTaxOn(double anAmount) {
    System.out.println("The tax for " + anAmount + " is: " +
anAmount
        * Order.taxRate);
  }

  public Order(MyDate d, double amt, String c, String p, int q)
{
    orderDate = d;
    orderAmount = amt;
    customer = c;
    product = p;
    quantity = q;
  }

  public String toString() {
    return quantity + " ea. " + product + " for " + customer;
  }

  public double computeTax() {
    System.out.println("The tax for this order is: " +
orderAmount
        * Order.taxRate);
    return orderAmount * Order.taxRate;
  }
```

```
}
```

MyDate.java

```
package com.acme.utils;

public class MyDate{
  // Member/instance variables   (a.k.a.
fields/properties/attributes)
  public int day;
  public int month;
  public int year;

  // Constructors:
  //   1. Same name as the class
  //   2. No return type

  //The no-args constructor
  public MyDate(){
  }

  //Constructor that takes 3 arguments
  public MyDate(int m, int d, int y){
    setDate(m, d, y);
  }

  //Methods
  public String toString(){
    return month + "/" + day + "/" + year;
  }
  public void setDate(int m, int d, int y){
    day = d;
    year = y;
    month = m;
  }
}
```

TestOrders.java

```
package com.acme.testing;
import com.acme.domain.Order;
import com.acme.utils.MyDate;

public class TestOrders {

  public static void main(String[] args) {
    MyDate date1 = new MyDate(1,20,2008);
    Order anvil = new Order(date1, 2000.00, "Wile E Coyote",
"Anvil",
       10);

    MyDate date2 = new MyDate(4,10,2008);
    Order balloons = new Order(date2, 1000.00, "Bugs Bunny",
       "Balloon", 125);

    System.out.println(anvil);
    System.out.println(balloons);

    System.out.println("The tax rate is currently: " +
Order.taxRate);
    Order.computeTaxOn(3000.00);
    anvil.computeTax();
    balloons.computeTax();

    Order.setTaxRate(0.06);
    System.out.println("The tax rate is currently: " +
Order.taxRate);
    Order.computeTaxOn(3000.00);
    anvil.computeTax();
    balloons.computeTax();
  }

}
```

TestMyDate.java

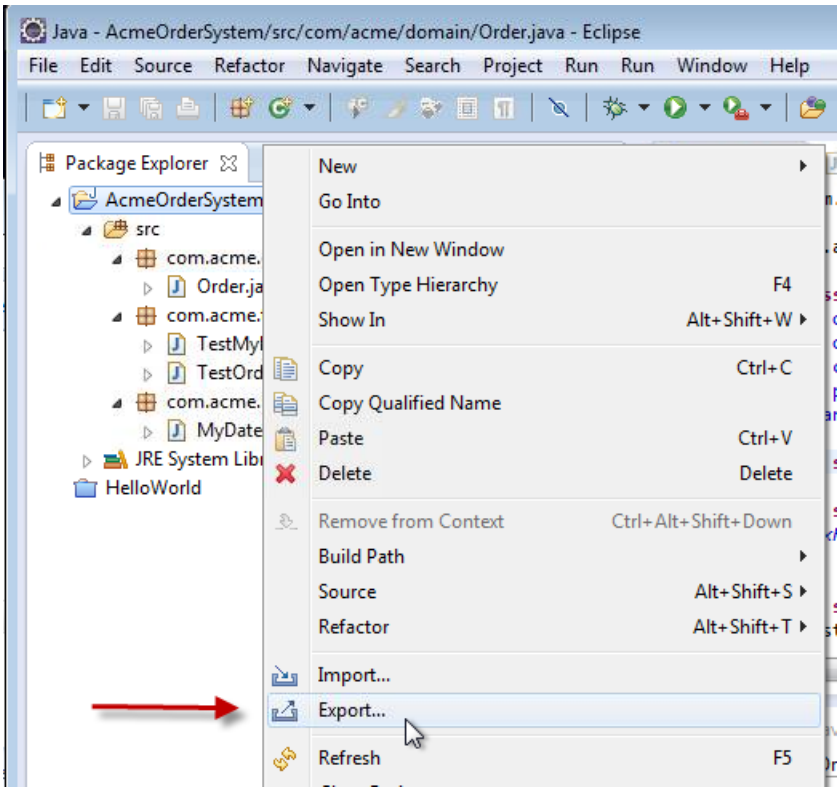
```
package com.acme.testing;
import com.acme.utils.MyDate;

public class TestMyDate{

  public static void main(String[] args){
    MyDate date1 = new MyDate(11,11,1918);

    MyDate date2 = new MyDate();
    date2.day = 11;
    date2.month = 11;
    date2.year = 1918;

    MyDate date3 = new MyDate();
    date3.setDate(4,21,1968);

    String str1 = date1.toString();
    String str2 = date2.toString();
    String str3 = date3.toString();

    System.out.println(str1);
    System.out.println(str2);
    System.out.println(str3);
  }
}
```

Bonus Lab

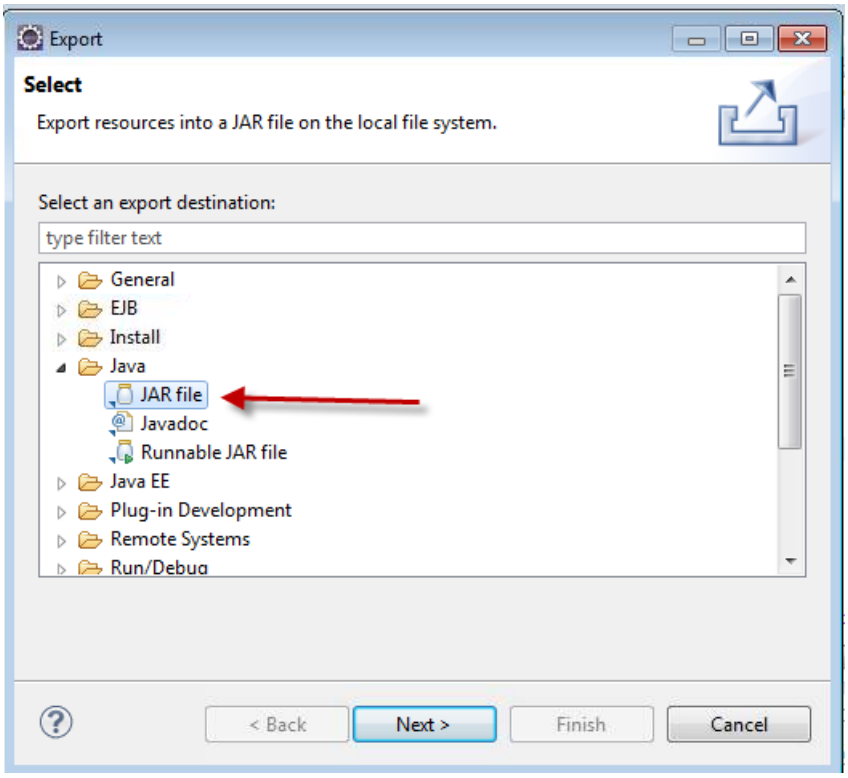
Step 5: Create a Java archive

Another team at Acme has found out about the wonderful job you are doing with the new order system. They would like to get your code to start experimenting with it, so you need to create a JAR file with your current code.

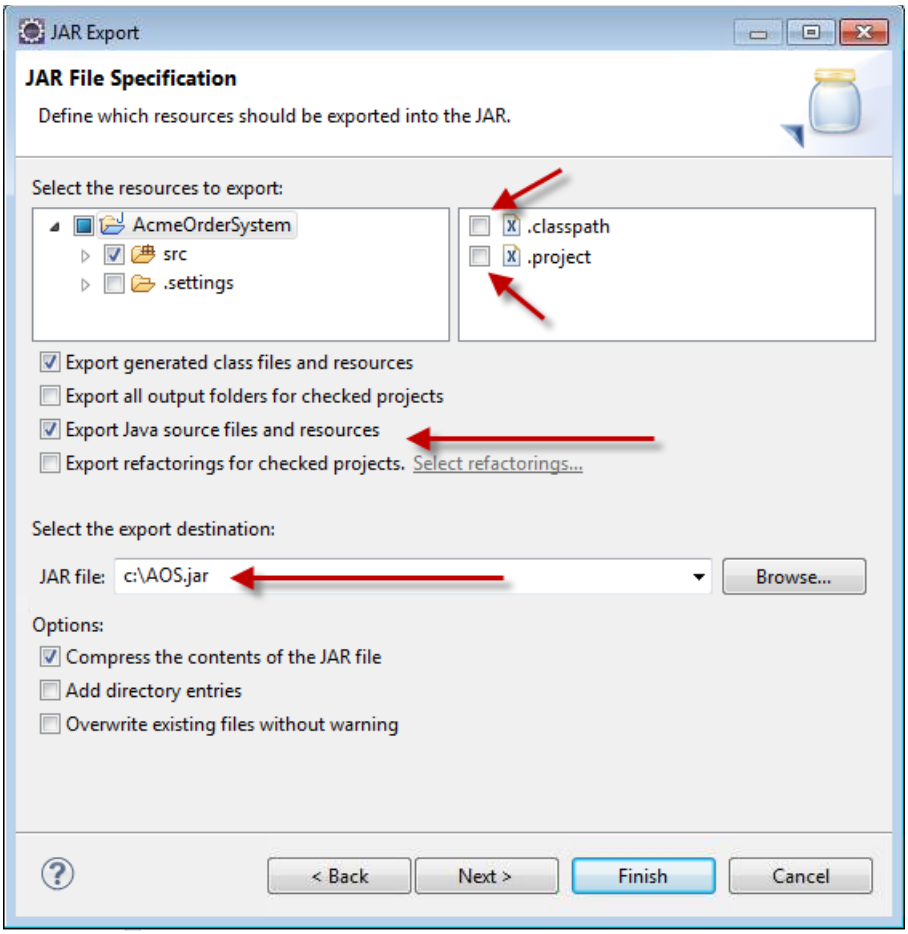
5.1 Export your project as a JAR file. Right-click on the AcmeOrderSystem project in the Package Explorer view and select ***Export...***.



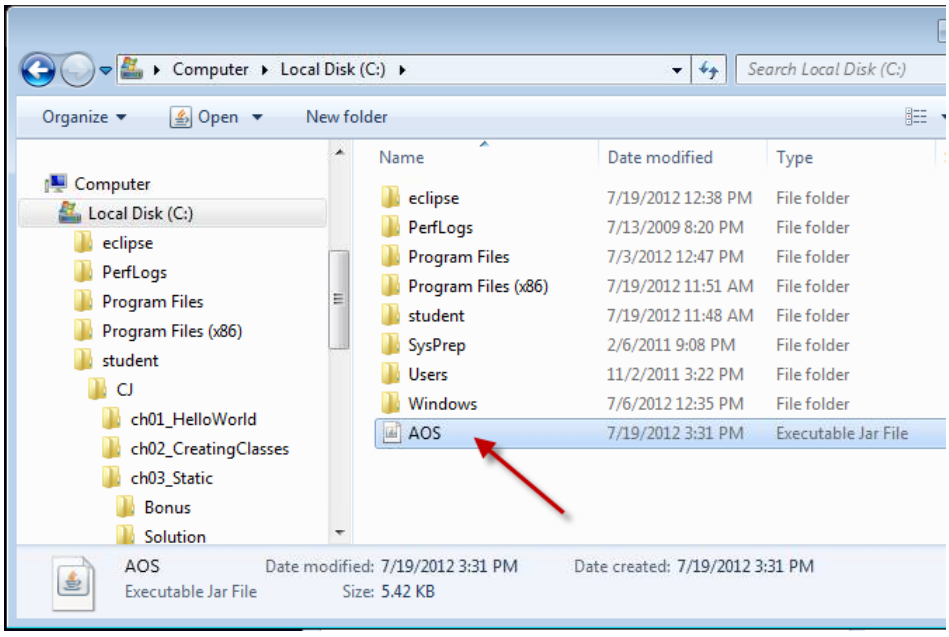
5.2 In the Export window, expand the Java type, select JAR file (as shown in the image below), and click the ***Next>*** button.



5.3 In the JAR Export window, uncheck the .classpath file and .project file. Check the Export Java source files and resources option and provide a location for the JAR file and click the ***Finish*** button.



5.4 In the destination location, you should find a JAR file. You can use a zip utility to open the file and ensure its contents include your project's Java source and class files.



Give Intertech a call at **1.800.866.9884** or visit Intertech's website.